Learning Queries for Relational, Semi-structured, and Graph Databases

Radu Ciucanu

Learning Queries for Relational, Semi-structured, and Graph Databases

Radu Ciucanu
University of Lille & INRIA, France
radu.ciucanu@inria.fr
Expected graduation date: 2015
Supervised by Angela Bonifati & Sławek Staworko

ABSTRACT

Web applications store their data within various database models, such as relational, semi-structured, and graph data models to name a few. We study learning algorithms for queries for the above mentioned models. As a further goal, we aim to apply the results to learning cross-model database mappings, which can also be seen as queries across different schemas.

Categories and Subject Descriptors

H.2.3 [**Database Management**]: Languages—*Query Languages*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept Learning*

General Terms

Algorithms, Languages, Theory.

Keywords

Query inference, XML, XPath, learning, twig query, natural join, semijoin, database mapping.

1. INTRODUCTION

Web applications may use relational, semi-structured, or graph databases to store their data. Sometimes, applications using different data models need to exchange data among them. The solutions for this kind of problem are typically based on defining *mappings*, which are logical assertions between elements in the source schema and elements in the target schema [1, 2, 7, 13, 14, 29]. Moreover, a mapping can be seen as a rule having (conjunctive) queries as its body and head. Typically, the mappings are defined by an expert user having exact knowledge of the schemas and semantics of the source and target databases.

An inherent research question is how to automatically infer schema mappings instead of asking an expert to define them. Recently, the problem of learning schema mappings has been studied for relational databases [25, 37, 38], but, to the best of our knowledge, no research has been done on learning cross-model database mappings. This motivates us to study the theoretical foundations of this problem. The approach that we propose consists in developing learning algorithms for queries for different data models. This means that in the process of data exchange, the user having exact knowledge of the source schema can be replaced by a learning algorithm, trained by a non-expert user. The query on the source database can thus be inferred from examples instead of being explicitly written.

The main objective of the thesis is to develop learning algorithms for queries for relational, semi-structured, and graph databases. As stated above, an important application of these algorithms is the automatic inference of mappings between different database models. We plan to design algorithms which intervene in the first phase of the process of cross-model data exchange, where, given some examples from the source database, the algorithms learn a query which extracts the data to be exchanged. In the second phase, based on techniques that still need to be investigated, there can be inferred the query which indicates how the extracted data must be incorporated into the target database.

Since the source and the target database do not necessarily use the same data model, we can imagine several scenarios of cross-model data exchange. We present four of them in Figure 1. For instance, if an application employing a relational database wants to *publish* [12, 16] (scenario 1) some of its data to make it available to an application using a semi-structured database (based on XML), the data should be first extracted with a relational query, such as a SQL query. In the other direction, if the application using a semi-structured database wants to *shred* [20] (scenario 2) some of its data to enrich a relational database, the data should be first extracted with a query over XML, for example with XPath or XQuery. The same query languages can be used to extract semi-structured data before shredding it into a graph database, based on RDF (scenario 3). Finally, scenario 4 illustrates extracting data from the graph database (for example with SPARQL) and then publishing it in XML format. Other pairs of heterogeneous data models are worth investigating (i.e., relational-to-graph), also due to the appearance of interoperability scenarios in the Semantic Web. We omit their discussion here for conciseness.

For each database model, we plan to study learning algorithms depending on several parameters. One parameter is the type of examples which are permitted, more precisely whether the algorithms take as input only positive or

SIGMOD'13 PhD Symposium, June 23, 2013, New York, NY, USA.

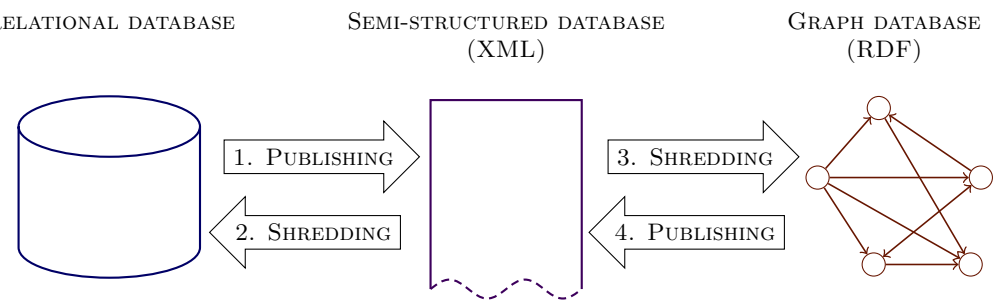

Figure 1: Data exchange between heterogeneous data models, using XML as intermediate model.

both positive and negative examples. Another parameter is whether there is a fixed set of examples or the learning protocol allows the algorithms to interactively ask the user to label more examples.

The structure of this paper reflects the two directions of research that we aim to investigate during the thesis. In Section 2 we present some ideas for extending the learnability of twig queries, while in Section 3 we talk about learning relational and graph queries. We present the conclusions in Section 4.

2. EXTENDING TWIG LEARNING

In the first stage of the thesis, we are interested in learning XML queries, which are the basis of shredding data stored in XML format into any other data model. Learning XML queries and transformations have been studied for the task of data extraction [21, 22], which is closely related. The classical framework of language inference in the limit [24] has been adapted to learn n-ary XML queries captured with tree automata [9, 28] and tree transformations captured with tree transducers [27]. While tree automata and transducers are valued for their ability to model large classes of queries, they have little support from the existing infrastructure which favors more common standards like XPath and XQuery.

Recently, Staworko and Wieczorek have investigated the learnability of twig queries [36], a highly practical and commonly used subclass of XPath. They have identified the subclass of anchored twig queries that are learnable from positive examples only, where the examples are XML documents with annotated nodes. They have shown, however, that adding negative examples renders learning more complex: it is NP-complete to decide whether there exists a query that selects all the positive examples and none of the negative ones. On the other hand, when considering the restriction that the sets of positive and negative examples have a bounded size, the problem becomes tractable [10]. In the setting of XML shredding, we may have informative negative examples and clearly the query learning approach should take advantage of it.

Since learning twig queries from positive and negative examples is intractable in general, we intend to study an approximate learning framework, such as PAC [40]. In this setting, the learned query may select some negative examples and omit some positive ones. We also plan to address the intractability of the consistency by considering richer query languages e.g., unions of twig queries for which testing consistency is trivial but learnability remains an open question.

We have evaluated the algorithms proposed in [36] on several XML benchmarks and we have observed that the algorithms are able to learn a query equivalent to the goal query from a small number of examples (generally two). The documents typically follow a certain schema that is often given in the form of a DTD or XML Schema, which consists of a set of constraints on the structure of the document. The experiments have pointed out that the learning algorithms may return *overspecialized queries*, which include fragments implied by the schema. The overspecialization occurs because the learning algorithms are based on the identification of all common patterns of the examples so the queries contain many conditions that follow from the schema of the documents. These conditions are always satisfied by the documents of the schema, making the returned query bigger and increasing its evaluation time.

To address the problem of overspecialization, an idea is to add the schema of the XML documents to the process of learning twig queries. Unfortunately, this reduces to fundamental decision problems, typically intractable: query satisfiability, query implication, and query containment in the presence of DTDs [5, 32]. In this context, we are interested in designing new XML schema formalisms, simpler than the DTDs, and in studying the complexity of the problems of interest related to them. Our research led to the *disjunctive multiplicity schema* and its restriction, the *disjunction-free multiplicity schema* [6]. These formalisms ignore the relative order among the elements from the XML documents. The motivation is that this order is not taken into account by the twig queries, hence is not important for solving problems such as query satisfiability or query containment in the presence of schema. Moreover, the disjunctive multiplicity schema can express the DTD from XMark [35], a well-known benchmark for XML data management. The disjunctive multiplicity schema also captures many of the DTDs from the real-world XML web collection proposed in [26].

We have performed a detailed static analysis of the problems of interest related to our schema formalisms. We have developed a set of tools to analyze the complexity upper bounds and we have shown that the lower bounds generally follow from the DTDs. A technical contribution is the polynomial algorithm for testing containment of two disjunctive multiplicity schemas. It is known that DTD containment is in PTIME when only 1-unambiguous regular expressions are allowed, PSPACE-complete for general regular expressions, and coNP-hard in the case of disjunction-free DTDs [31]. For disjunction-free multiplicity schemas, we have reduced query satisfiability and query implication to testing embedding from the query to some dependency

graphs, so we can decide them in PTIME. The set of tools that we have developed around the dependency graphs also permitted us to analyze the complexity of query implication and query containment in the presence of disjunction-free DTDs. To the best of our knowledge, these problems were not previously studied in the literature and we have proved that they are in PTIME and coNP-complete, respectively. Query containment in the presence of DTDs is EXPTIME-complete [32] and although we obtain a considerably lower complexity for the restriction of disjunction-free multiplicity schema (coNP-completeness), the problem remains intractable. The research that we have done on schemas for unordered XML pointed out that adding the schema of the documents to the process of learning twig queries reduces to a problem which is intractable even for a schema less expressive than the DTDs. One possible continuation of this direction is to explore within an approximate learning framework the idea of taking into account the schema of the documents in the process of query learning.

On the other hand, we are interested in exploring query optimization techniques based on the schema of the documents. This might lead to an optimized version of the algorithms from [36]. The difference is that we want to add a filter present in all the positive examples to the learned query only if it is not implied by the schema. As we have already pointed out, the problem of query implication can be decided in PTIME for the disjunctive multiplicity schema because it is equivalent to testing the embedding of the query in a dependency graph [6]. In the optimized version, when we add a filter to the learned query, we know that the filter is not implied by the schema, but we do not know whether the query with the filter is equivalent in the presence of schema with the same query without the filter. The optimization that we propose is of interest because query implication is a tractable problem, while query containment is not. The schema of the documents can be given explicitly or inferred from the documents on which the user selects nodes to train the algorithm. In the latter case, the schema must be learned from positive examples only and our preliminary research pointed out that the disjunctive multiplicity schemas are identifiable in the limit [24] from positive examples only.

Finally, we plan to evaluate the capabilities of our algorithms w.r.t. XMark [35]. There are 20 proposed queries for XML documents generated under XMark. These queries are in XQuery and only 10 over the 20 can be expressed in XPath. Franceschet [19] proposed XPathMark, an XPath benchmark on top of the XMark generated data. The algorithms from [36] are able to learn 15% of the queries from XPathMark. We want to test the optimized version of the algorithms from [36] on these queries. We can measure the size of the learned query before and after adding the schema to the learning process and observe with what percentage the size decreases when the schema is involved. We also want to develop a practical system able to learn twig queries from interaction with the user.

3. LEARNING RELATIONAL AND GRAPH QUERIES

In the second stage of the thesis, we want to explore the inference of relational and graph queries. In the case of relational queries, we plan to concentrate on simple operators, such as join-like operators. We want to apply this kind of queries to extract data from a relational database before publishing it in XML or RDF format. In the sequel, we present the related work, the framework that we want to investigate, extensions of the approach to graph databases, and a possible application to crowdsourcing databases.

Bancilhon [3] and Paredaens [33] studied the decision problem, given a pair of relational instances, whether there exists a relational algebra expression which maps the first instance to the second one. Their research led to the notion of *BP-completeness*. Their results were later extended to the nested relational model [41] and to sequences of input-output pairs [18]. A related problem, recently studied by Tran et al. [39], is the *query by output* problem: given a database instance and the output of some query, their goal is to construct an instance equivalent query to the initial one. Das Sarma et al. [11] investigated the *view definitions problem*: given a database instance and a corresponding view instance, find the most succinct and accurate view definition. Fan et al. [15] proposed learning algorithms for conditional functional dependencies. The last three mentioned contributions use data mining techniques. In a different approach, Gillis and Van den Bussche [23] used inductive logic programming to infer relational algebra expressions.

We plan to focus our research on a learning setting which is slightly different from those mentioned above. We assume a very large database instance and a user giving annotations on it. We propose an *interactive framework* where our learning algorithms choose tuples and then ask the user to label them as positive or negative examples. After each label given by the user, our algorithms infer the tuples which become *uninformative* w.r.t. the previously labeled tuples. The interactive process stops when all the tuples in the instance either have a label explicitly given by the user, or they have become uninformative. The output of the algorithm is a query consistent with the examples i.e., which selects all the positive examples and none of the negatives. The goal is to minimize the number of interactions with the user.

We performed preliminary research in this direction, on learning two types of relational queries: natural joins and semijoins. For the *natural joins*, we have proved the tractability of some problems of interest, such as testing consistency of a set of positive and negative examples, a problem which is intractable in the context of *semijoins*. We want to extend our approach to other operators and also to chains of joins between many relations. Besides the goal of minimizing the number of interactions with the user, we are also interested in designing strategies feasible in practice. This implies optimizations at two different levels. First of all, for the relational queries which have a polynomial consistency checking for positive and negative examples (e.g., natural joins), we have to design algorithms which efficiently explore the potentially exponential space of possible options at each step in the interactive process. On the other hand, in the case of relational queries for which consistency checking is intractable for positive and negative examples (e.g., semijoins), the problem is even harder. This case is similar to learning twigs from positive and negative examples (as explained in Section 2), which motivates us to also investigate an approximate learning framework. Then, also for this type of relational queries, the goal is to design strategies minimizing the number of interactions with the user. The difference is that some of the annotations might be ignored to be able to compute in polynomial time a candidate query.

Next, we want to investigate a similar approach in the context of graph databases, which gained popularity during the last years, being involved in many emerging Semantic Web applications. Learning graph queries represents the basis of learning schema mappings for exchanging data between graph databases and any other data model. Schema mappings and data exchange in the context of graph databases are novel research topics [8]. These problems were very recently studied by Barceló et al. [4], who propose mapping languages based on the most typical graph database queries, such as regular path queries and conjunctions of nested regular expressions. We aim to identify a query language for graphs which is expressive enough and also learnable from positive and possibly negative examples. In the context of RDF, the standard querying language is SPARQL, which, unfortunately, is too expressive and involves too computationally complex problems for our purposes. More precisely, the evaluation of general SPARQL patterns is PSPACE-complete, while the evaluation of the restricted class of "well-designed" patterns is coNP-complete [34]. The navigational query languages for graphs were studied in [17]. To the best of our knowledge, no research has been done on learning graph queries or on learning schema mappings from graph databases to any other data model. We want to explore an interactive learning framework, as already explained for the relational queries.

To illustrate a possible use case of such an application, take for instance a geographical database modeled as a graph. The vertices represent cities and the edges store information such as the distance between the cities, the type of road linking the cities (e.g., highway), etc. Obviously, such a database might be very large since it may store information about all the cities from a country or a continent. The user is interested in extracting paths between cities i.e., sets of vertices and the edges linking them. First, the user has to select two vertices from the graph, so the two cities which are the extremities of the path. But she certainly is not interested in all the paths linking the two cities, since their number might be considerable. The user may also want to impose certain restrictions on the paths, such as the total distance, the type of road, or an intermediate city on the path. Our algorithms compute what paths the user should be asked to label (as positive or negative example) in order to gather as many information as possible with few interactions. Additionally, the learning framework must be able to use query workload techniques to take advantage of the previously inferred paths. For instance, consider a scenario where all the previous users were interested in paths where all the edges linking the two extremity cities contain the information "highway" as type of road. In this case we want to ask with priority the next user to label a path having the same property. Finally, when the user is satisfied with the proposed paths, the interactive loop stops and the extracted data can be then published as XML, inserted in a relational database, or transfered to another graph, depending on the concrete target database schema.

An interactive framework for query learning can be also useful in the context of crowdsourcing databases. Recently, Marcus et al. [30] have investigated how to use human input to compare items for joining data. They propose a system which, for example, optimizes the overall cost of joining two sets of images. Their and our approach have in common the goal of minimizing the number of interactions with the user, which, in their case, are paid workers. Such an interaction is called Human Intelligence Task (HIT) in terms of crowdsourcing marketplaces and involves an employer who pays a certain amount of money to workers to solve it. A consequence is that for the crowdsourcing applications, minimizing the number of interactions with the user is equivalent to minimizing the financial cost of the process. The difference between their and our problem is that we know the relations of the tables from the instance since the beginning, while they are interested in using "features" for filtering purposes only in an optimized version of their system. The features are essentially attributes of the photos which are inferred also as HITs, therefore against a financial cost. The set of features has been proposed to motivate the choice of which attributes to join in crowdsourcing scenarios. Such techniques can also be applied to our context to decide which tuples can be proposed to the user to make the learning process more effective. Therefore, the inference of joins in the context of crowdsourcing databases can be seen not only as another potential application of our algorithms, but also as a source of inspiration for designing strategies efficient in practice for our initial problem.

4. CONCLUSIONS

The main goal of the thesis is to develop learning algorithms for queries for relational, semi-structured, and graph databases. We want to use our algorithms to address the problem of learning cross-model database mappings, which is novel in the literature. In the process of cross-model data exchange, our contributions can be used to automate the first stage of the process i.e., extracting the data from the source database before transferring it to the target database.

The algorithms that we aim to propose can be applied to replace the expert user having complete knowledge of the source database schema with a learning algorithm trained by a non-expert user. Since the source database instance may have a significant size, our challenge is to propose learning algorithms which minimize the annotations that the user must provide. For the second stage of the data exchange process i.e., incorporating the extracted data into the target database, the learnability problem remains open.

There already exist learning algorithms for XML queries, particularly for the classes of path and twig queries [36], which are in fact restrictions of XPath. In this case our goal is to take advantage of the existing techniques, enrich the learning framework to allow both positive and negative examples, and also take into account the schema of the documents. In terms of practical results, we aim to validate our algorithms with queries from XPathMark. The algorithms proposed in [36] have the nice property that they are able to learn the goal query from very few examples, therefore they require little information given by the user, and we definitely want to preserve this property in our extensions.

On the other hand, for relational and graph queries, the solutions already proposed in the literature do not really correspond to the framework that we want to explore. Consequently, for the queries for these models we have to investigate from scratch the interactive learnability from positive and negative examples. Finally, we plan to evaluate if the learning algorithms are efficient in practice and also expressive enough w.r.t. adequate benchmarks for both relational and graph databases.

5. REFERENCES

[1] B. Alexe, L. Chiticariu, R. Miller, and W. C. Tan. Muse: Mapping understanding and design by example. In *ICDE*, pages 10–19, 2008.

[2] M. Arenas and L. Libkin. XML data exchange: Consistency and query answering. *J. ACM*, 55(2), 2008.

[3] F. Bancilhon. On the completeness of query languages for relational data bases. In *MFCS*, pages 112–123, 1978.

[4] P. Barceló, J. Pérez, and J. Reutter. Schema mappings and data exchange for graph databases. In *ICDT*, pages 189–200, 2013.

[5] M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of DTDs. *J. ACM*, 55(2), 2008.

[6] I. Boneva, R. Ciucanu, and S. Staworko. Simple schemas for unordered XML. `http://arxiv.org/abs/1303.4277`, 2013.

[7] A. Bonifati and Y. Velegrakis. Schema matching and mapping: from usage to evaluation. In *EDBT*, pages 527–529, 2011.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Simplifying schema mappings. In *ICDT*, pages 114–125, 2011.

[9] J. Carme, R. Gilleron, A. Lemay, and J. Niehren. Interactive learning of node selecting tree transducer. *Machine Learning*, 66(1):33–67, 2007.

[10] S. Cohen and Y. Weiss. Certain and possible XPath answers. In *ICDT*, pages 237–248, 2013.

[11] A. Das Sarma, A. Parameswaran, H. Garcia-Molina, and J. Widom. Synthesizing view definitions from data. In *ICDT*, pages 89–103, 2010.

[12] A. Deutsch and V. Tannen. MARS: A system for publishing XML from mixed and redundant storage. In *VLDB*, pages 201–212, 2003.

[13] R. Fagin, L. Haas, M. Hernández, R. Miller, L. Popa, and Y. Velegrakis. Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*, pages 198–236, 2009.

[14] R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[15] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Trans. Knowl. Data Eng.*, 23(5):683–698, 2011.

[16] M. Fernandez, Y. Kadiyska, D. Suciu, A. Morishima, and W. C. Tan. SilkRoute: A framework for publishing relational data in XML. *ACM Trans. Database Syst.*, 27(4):438–493, 2002.

[17] G. Fletcher, M. Gyssens, D. Leinders, J. Van den Bussche, D. Van Gucht, S. Vansummeren, and Y. Wu. Relative expressive power of navigational querying on graphs. In *ICDT*, pages 197–207, 2011.

[18] G. Fletcher, M. Gyssens, J. Paredaens, and D. Van Gucht. On the expressive power of the relational algebra on finite sets of relation pairs. *IEEE Trans. Knowl. Data Eng.*, 21(6):939–942, 2009.

[19] M. Franceschet. XPathMark: An XPath benchmark for the XMark generated data. In *XSym*, pages 129–143, 2005.

[20] J. Freire and J. Siméon. Adaptive XML shredding: Architecture, implementation, and challenges. In *EEXTT*, pages 104–116, 2002.

[21] R. Gilleron, F. Jousse, I. Tellier, and M. Tommasi. XML document transformation with conditional random fields. In *INEX*, pages 525–539, 2006.

[22] R. Gilleron, P. Marty, M. Tommasi, and F. Torre. Interactive tuples extraction from semi-structured data. In *Web Intelligence*, pages 997–1004, 2006.

[23] J. Gillis and J. Van den Bussche. Induction of relational algebra expressions. In *ILP*, pages 25–33, 2009.

[24] E. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[25] G. Gottlob and P. Senellart. Schema mapping discovery from data instances. *J. ACM*, 57(2), 2010.

[26] S. Grijzenhout and M. Marx. The quality of the XML web. In *CIKM*, pages 1719–1724, 2011.

[27] A. Lemay, S. Maneth, and J. Niehren. A learning algorithm for top-down XML transformations. In *PODS*, pages 285–296, 2010.

[28] A. Lemay, J. Niehren, and R. Gilleron. Learning n-ary node selecting tree transducers from completely annotated examples. In *ICGI*, pages 253–267, 2006.

[29] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[30] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[31] W. Martens, F. Neven, and T. Schwentick. Complexity of decision problems for XML schemas and chain regular expressions. *SIAM J. Comput.*, 39(4):1486–1530, 2009.

[32] F. Neven and T. Schwentick. On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. *Logical Methods in Computer Science*, 2(3), 2006.

[33] J. Paredaens. On the expressive power of the relational algebra. *Inf. Process. Lett.*, 7(2):107–111, 1978.

[34] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.

[35] A. Schmidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, and R. Busse. XMark: A benchmark for XML data management. In *VLDB*, pages 974–985, 2002.

[36] S. Staworko and P. Wieczorek. Learning twig and path queries. In *ICDT*, pages 140–154, 2012.

[37] B. Ten Cate, V. Dalmau, and P. Kolaitis. Learning schema mappings. In *ICDT*, pages 182–195, 2012.

[38] B. Ten Cate, P. Kolaitis, and W. C. Tan. Schema mappings and data examples. In *EDBT*, pages 777–780, 2013.

[39] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *SIGMOD Conference*, pages 535–548, 2009.

[40] L. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[41] D. Van Gucht. On the expressive power of the extended relational algebra for the unnormalized relational model. In *PODS*, pages 302–312, 1987.